VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Dashboard pro mobilní platformy

Dashboard for Mobile Platforms

2016 Martin Cejpek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Cejpek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Dashboard pro mobilní platformy
Dashboard for Mobile Platforms

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit multiplatformní mobilní aplikaci (s využitím např. PhoneGap), která bude umožňovat obecné zobrazování dat formou dashboardu. Data budou poskytována serverem a úkolem aplikace bude jejich vizualizace, nastavení zobrazení, aktualizace, apod.

1. Popište možnosti vývoje multiplatformních mobilních aplikací.
2. Navrhněte architekturu aplikace pro vizualizaci dat na základě dat obdržených ze serveru. Bude se typově jednat o provozní data, statistická data, apod.
3. Implementujte aplikaci v prostředí umožňující multiplatformní nasazení.
4. Zhodnoťte výslednou aplikaci a možnosti nasazení na konkrétní platformy.

Seznam doporučené odborné literatury:

[1] Mark Lassoff: Mobile App Development with HTML5, LearnToProgram, 2015, ISBN: 978-0692405055
[2] Wilkins Fernandez: Beginning App Development with Parse and PhoneGap, Apress, 2015, ISBN: 978-1484202364
[3] Cameron Banga:
Essential Mobile Interaction Design: Perfecting Interface Design in Mobile Apps (Usability), Addison-Weslley Professional, 2014, ISBN: 978-0321961570

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 01.09.2015
Datum odevzdání: 29.04.2016

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA 1D16

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 .....

Rád bych poděkoval všem, kteří mi zapůjčili potřebnou literaturu a nebo mě jakkoli podpořili při psaní této bakalářské práce.

Abstrakt

Tato bakalářská práce s názvem „Dashboard pro mobilní platformy“ se zabývá problematikou vývoje dashboardové aplikace pro různé operační systémy. Zejména se zde řeší problematika pokrytí více operačních systémů při vývoji aplikace, typy mobilních aplikací a dostupné vývojářské nástroje. Práce se věnuje především vytváření aplikace pomocí vývojářského nástroje PhoneGap. Také se zde rozebírá architektura aplikace a způsob, jakým bude aplikace data získávat a zobrazovat. Práce se ke konci zabývá testováním a nasazením aplikace a jejím hodnocením.

Klíčová slova: Cordova, dashboard, Google Analytics, hybridní aplikace, chytrý klient, mobilní aplikace, multiplatformní aplikace, PhoneGap

Abstract

This bachelor thesis with title „Dashboard for Mobile Platforms“ deals with the issue of developing mobile dashboard application for various opereating systems. In particular it unravels the issue of covering multiple operating systems during the development of an application, the types of mobile applications and the available developer tools. The thesis addresses mainly the development of the application using PhoneGap framework. The architecture of the application and the way it gets and displays data is analyzed as well. By the end the thesis deals with testing, deployment and evaluation of the application.

Key Words: Cordova, dashboard, Google Analytics, hybrid application, smart-client, mobile application, multiplatform application, crossplatform application, PhoneGap

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
CLI	–	Command-line Interface
CSS	–	Cascading Style Sheet
GUI	–	Graphical User Interface
HTML	–	Hyper Text Markup Language
IP	–	Internet Protocol
JSON	–	JavaScript Object Notation
QR	–	Quick Response
RESTful	–	Representational State Transfer
SDK	–	Software Development Kit
Symlink	–	Symbolic Link
URL	–	Uniform Resource Locator
UI	–	User Interface
XML	–	Extensible Markup Language

Seznam obrázků

1 Úvod

V posledních několika letech zažívají chytré telefony obrovský rozmach v popularitě. Dle statistik na stránkách www.statista.com bylo v roce 2007 prodáno 122,32 miliónů chytrých telefonů, v roce 2011 to bylo 472 miliónů a v roce 2015 už 1,42 miliard. Za takový boom může především jejich stále klesající cena a zlepšující se hardwarové a softwarové vymoženosti. Díky tomu jsou tato zařízení nyní přístupnější širší veřejnosti. Lidé nyní mohou na svých mobilních zařízeních sledovat videa, poslouchat hudbu, vytvářet fotky s vysokým rozlišením a nebo se připojit k internetu.

Mnoho firem, jako například Apple, Google a Microsoft, se okamžitě chopilo šance a na trh uvedlo svou řadu chytrých telefonů. Mnoho vývojářů začalo využívat vymožeností chytrých telefonů pro vývoj a prodej jejich vlastních aplikací. Dnes již existují aplikace úplně na všechno - ať už se jedná o aplikaci pro objednávání jídla nebo pro vytváření rezervace u kadeřníka, lidé očekávají, že pro cokoliv nějaká aplikace existuje. Není se proto čemu divit, že se mobilní aplikace staly součástí našeho každodenního života.

Ovšem vývojáři mobilních aplikací čelí jednomu velkému problému. Operační systémy na mobilních zařízeních nejsou kompatibilní s ostatními. Stejně jako aplikace z MacBooku nejde spustit ve Windows, není možné spustit aplikaci vytvořenou pro iOS v mobilu s operačním systémem Androidu. Tento problém se snaží vyřešit tzv. multiplatformní aplikace, které umožňují aplikaci se stejným kódem zprovoznit na různých operačních systémech.

Tato práce se bude zabývat vývojem multiplatformní dashboardové aplikace. Dashboardovou aplikací se rozumí aplikace, která organizuje a graficky zobrazuje data ve snadno pochopitelné podobě. V tomto případě bude dashboardová aplikace zobrazovat statistiky stránek Katedry Informatiky Vysoké školy Báňské, technické univerzity Ostrava, pořízené službou Google Analytics.

První část této práce se zabývá možností vyvíjení multiplatformní aplikace v prostředí PhoneGap. Ve druhé části se rozebírá architektura vyvíjené aplikace. Ve třetí části se řeší jak se budou zpracovávat data ze serveru. Ve čtvrté části se přechází do fáze testování a debugování aplikace. V páté části se řeší možnosti nasazení a v poslední šesté části se výsledná aplikace hodnotí.

2 Vývoj mobilní aplikace

Tato část se zabývá procesem vývoje multiplatformní mobilní aplikace. Vysvětluje, jaké jsou typy mobilních aplikací, co to je multiplatformní aplikace a proces jejího vývoje. Nakonec se bude podrobně řešit vyvíjení aplikace v prostředí PhoneGap.

Protože hardwarové možnosti dnešních chytrých telefonů jsou velké, není se čemu divit, že se mnoho vývojářských týmů zaměřuje právě na vývoj aplikací pro tato zařízení. Chytrým telefonem se myslí zařízení, které má pokročilý operační systém, pokročilé funkce jako wi-fi, přehrávání videí, dotykový displej s vysokým rozlišením a jiné.

Problémem pro vývojáře aplikací pro chytré telefony je samotný operační systém zařízení. Každý výrobce chytrých telefonů používá vlastní operační systém, který není kompatibilní s operačními systémy jiných výrobců. Operační systém (zkráceně OS, nebo také „platforma“) spravuje hardwarové a softwarové prostředky zařízení. Například Google Android a Apple iOS jsou operačními systémy. Dalším problémem je, že každý operační systém používá u svých aplikací jiný programovací jazyk, například Android používá jazyk Java, zatímco Windows Phone 8 používá `C#` a `C++`. Problém různých OS tedy pro vývojáře znamená vytváření jedné aplikace v různých vývojových prostředích, v různých programovacích jazycích a kalibrovat je pro každou platformu zvlášť. Tento problém se snaží vyřešit tzv. multiplatformními aplikacemi.

2.1 Multiplatformní aplikace

Multiplatformní aplikace se drží paradigmatu „napsat jednou a spouštět kdekoliv“ a je tím myšleno právě odstranění problému s potřebou různých programovacích jazyků a vývojových prostředí při vyvíjení aplikace pro mnoho operačních systémů. Multiplatformní aplikace nejsou zrovna nový koncept vývoje aplikací, nicméně až v posledních letech se nabídla možnost vývoje kvalitních multiplatformních aplikací. V dnešní době se však nejedná o jedinou možnost vývoje aplikací, jde spíše o business strategii.

Prvním krokem při vývoji Dashboardové aplikace je určení typu aplikace. V dnešní době se nabízí následující tři typy mobilních aplikací:

- Nativní aplikace,
- webové aplikace,
- hybridní aplikace.

2.1.1 Nativní aplikace

Nativní aplikací se myslí aplikace, která využívá technologie patřící přímo danému operačnímu systému. Jedná se například o operační systémy Android, Blackberry, iOS nebo Windows Phone. Nativní aplikace pro Android jsou psány v jazyce Java, pro iOS je to jazyk Objective C a nebo jazyk Swift. Taková aplikace poté musí být schválena firmou, pro jejíž platformu byla vyvinuta,

aby ji firma umístila do svého obchodu s aplikacemi - například pro iOS to je App Store a u platformy Android se jedná o Google Play. Z těchto obchodů si ji uživatel může stáhnout s vědomím, že jsou dodrženy určité standardy mobilních aplikací jako je rychlost, náročnost nebo bezpečnost.

Výhodou nativní aplikace je její výkonnost a spolehlivost, protože snadno využívá software a zabudované hardwarové funkce daného zařízení. Nicméně nativní aplikace s sebou nesou i mnoho nevýhod. Pokud má aplikace oslovit širokou veřejnost, je zapotřebí ji vyvinout úplně od začátku zvlášť pro každou platformu, protože každá používá jiný programovací jazyk. Tento problém se poté u aplikace projevuje na její ceně, protože je zapotřebí najmout několik týmů vývojářů, kde každý tým vyvíjí a udržuje aplikaci pro jiný operační systém. Projevuje se i na časové náročnosti a aktualizacích, které jsou pro různé OS vydány v různé dny.

2.1.2 Webové aplikace

Webové aplikace jsou ve skutečnosti webové stránky, které byly předělány tak, aby vypadaly a fungovaly jako nativní aplikace, i když je jejich implementace naprosto odlišná. Webové aplikace využívají standardní webové technologie jako je HTML5, CSS a JavaScript pro své fungování a spouští se ve webovém prohlížeči. Přistupuje se k nim jako ke kterékoliv jiné webové stránce pomoci speciálního URL a nebo jejich nainstalováním na domácí obrazovku zařízení tak, že se uloží do záložek.

Výhodou těchto aplikací je, že doba a cena jejich vývoje je nízká, protože stačí napsat pouze jednu verzi kódu, který bude fungovat na všech platformách, a nevyžadují žádné místo v paměti zařízení.

Menší nevýhodou je, že je někdy potřeba aplikaci doladit pro každý operační systém jinak, ale množství práce na odstranění tohoto problému je naprosto minimální, oproti programování aplikace od začátku jako je tomu u nativních aplikací.

Hlavními problémy u těchto aplikací ale je, že neexistuje žádné centrální místo, kde by se aplikace dala koupit jako je tomu u nativních aplikací. Navíc webové aplikace nikdy nebudou tak výkonné, protože JavaScriptový překladač nikdy nebude stejně rychlý, jako už přeložený kód nativní aplikace. Velkou nevýhodou je, že nemá přístup k funkcím zařízení, protože se jedná o aplikaci ve webovém prohlížeči, a vývojáři se tedy musí často spokojit s nějakým horším řešením.

Webové aplikace představovaly první pokus o multiplatformní aplikace, ale právě kvůli jejich nízké kvalitě se nikdy na trhu příliš neuchytily.

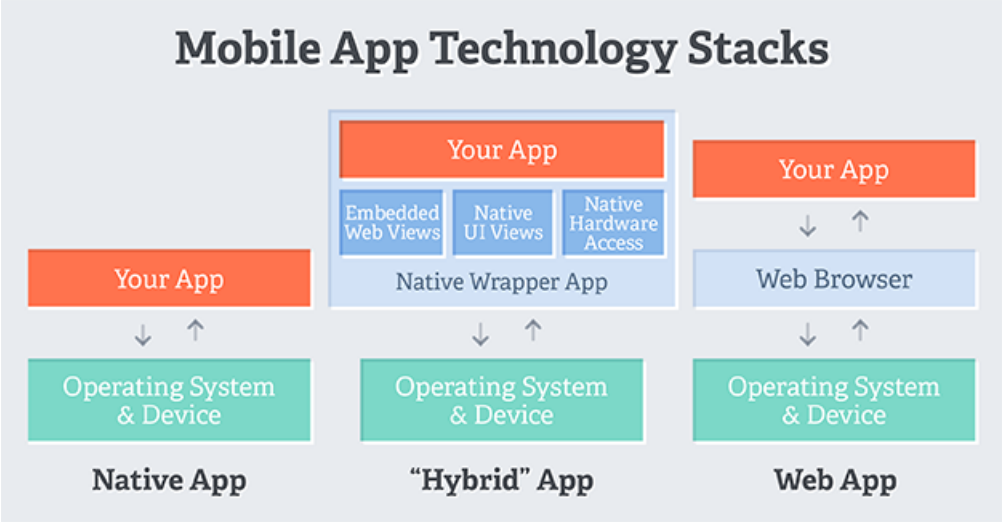
2.1.3 Hybridní aplikace

Hybridní aplikace jsou částečně nativní a částečně webové. Připomínají webové aplikace v tom, že využívají HTML, CSS a Javascript kód, který vykreslují ve webovém prohlížeči, ale samotný prohlížeč je zabalen v nativní aplikaci. Zabalením aplikace se myslí přidání nativního kódu, aby aplikace mohla na daném operačním systému fungovat jako standardní nativní aplikace.

Tento přístup má hned několik výhod. Vývoj aplikace je rychlý a levný, protože jádro aplikace (HTML, JS a CSS) zůstává stále stejné pro všechny platformy a pouze malá část nativního kódu se musí přepsat. Protože je aplikace zabalena v nativní aplikaci, je podobně jako „čisté“ nativní aplikace v aplikačním obchodu a může využívat funkce zařízení, na kterém běží.

Na druhou stranu hybridní aplikace nikdy nedosáhnou stejné rychlosti jako nativní aplikace, protože jsou stále závislé na rychlosti prohlížeče. Také nikdy nenabídnou stejně dobrý prožitek pro uživatele jako mohou nabídnout nativní aplikace - nikdy nevypadají tak přirozeně.

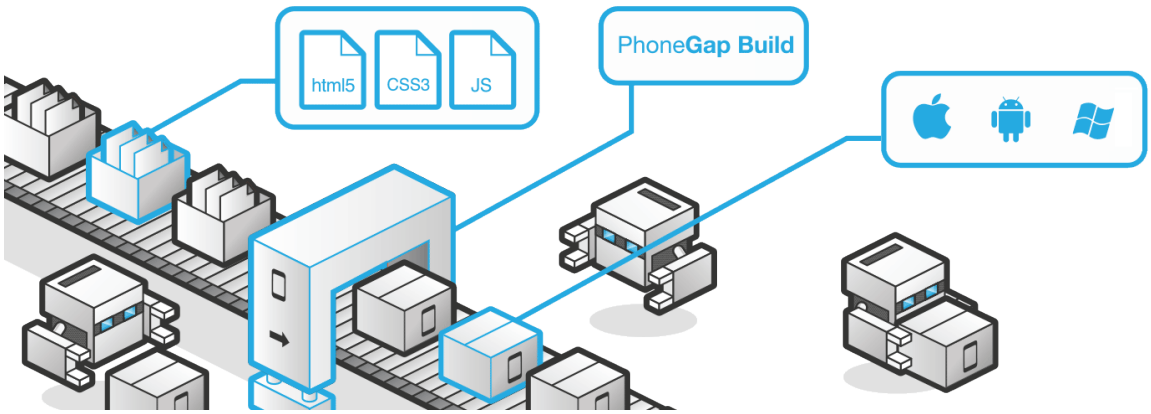
Obrázek 1 graficky zobrazuje strukturu jednotlivých typů aplikací. Nativní aplikace přímo komunikuje s operačním systémem a mobilním zařízením, ale za cenu komplikovaného vývoje. Webové aplikace mají naopak velmi snadný vývoj, ale s platformou komunikují pouze přes webový prohlížeč, a tím mají velmi omezené funkce. Hybridní aplikace jsou webové stránky zabalené v nativní aplikaci. Díky tomu mohou komunikovat přímo s platformou a přitom stále používat jeden a ten stejný kód.



Obrázek 1: Struktura nativních, hybridních a webových aplikací
Zdroj: https://myshadesofgray.files.wordpress.com/2014/04/mobile-app-tech-stacks.png

mobile-app-tech-stacks Hybridní aplikace se nabízí jako nejlepší volba pro vývoj multiplatformní aplikace především pro jejich nízkou cenu, rychlý vývoj a možnost používat je na různých platformách.

Dashboardová aplikace tedy bude vyvíjena jako webová aplikace - strukturu stránky bude definovat HTML soubor, aplikační logika bude řízena javascriptem a vzhled bude definován pomocí kaskádových stylů. Tyto webové stránky se poté nechají obalit pomocí aplikačního rámce (frameworku), aby fungovaly jako nativní aplikace pro danou platformu. Aplikační rámec sám řeší, jak aplikaci obalit. Díky tomu usnadní vývojáři práci, protože mu stačí napsat kód pouze jednou a poté ho zabalit pro různé platformy. Obrázek 2 tento postup graficky znázorňuje. Nyní je potřeba vybrat vhodný vývojářský nástroj.



Obrázek 2: Vytváření multiplatformních aplikací
Zdroj: https://build.phonegap.com/images/marketing/build-diagram.png

2.2 Vývojářské nástroje

Na trhu existuje hned několik vývojářských nástrojů pro multiplatformní mobilní aplikace. Mezi nejznámější patří Xamarin, který umožňuje vytvářet nativní aplikace v jazyce založeném na jazyku `C#`. Jedná se o výjimku v oblasti tvorby hybridních aplikací, protože nepoužívá webovou technologii.

Dalším známým nástrojem je Appcelerator, který využívá JavaScript. Vývojářům nabízí zabudované analytické nástroje pro rychlejší vývoj a testování aplikací a virtuální soukromé cloudové uložiště.

Především pro firmy je známý nástroj iFactr. iFactr nabízí uživatelům možnost rychlého vytváření aplikací, nízkou křivku zaučení a robustní prototypovací funkce.

V neposlední řadě tu je PhoneGap, který vývojářům umožňuje vytvářet aplikace z HTML, CSS a JavaScript souborů, aniž by výsledné aplikace ztratily funkčnost nativních aplikací. Dashboardová aplikace bude vyvíjena právě pomocí nástroje PhoneGap.

2.3 PhoneGap

PhoneGap je vývojářský nástroj od společnost Adobe, který umožňuje postavit hybridní aplikace využitím CSS3, HTML5 a JavaScriptu. PhoneGap byl nejprve vytvořen firmou Nitobi Software, kterou v roce 2011 zakoupil Adobe. Zdrojový kód PhoneGapu byl následně předán do Apache Software Foundation, kde byl použit k založení nového open-source[1] projektu s názvem Apache Cordova. Nyní je PhoneGap prodejní verzí a systémem nad Apache Cordova.

PhoneGap je pouze médiem pro předělání webové aplikace v hybridní mobilní aplikaci. Nejedná se o vývojářský nástroj určený pro psaní webových aplikací, jako je například NetBeans, Eclipse či Microsoft Visual Studio. PhoneGap pouze nabízí vytvoření a správu projektu, testo-

[1] Open-source projekt je projekt se zveřejněným zdrojovým kódem

vání a debugování vyvíjené aplikace a nakonec předělání projektu v hybridní aplikaci pro zvolený operační systém.

PhoneGap nabízí mnoho nástrojů pro vývoj aplikací. Vývoj může probíhat lokálně na počítači nebo vzdáleně využitím PhoneGap Build. Pro lokální vývoj jsou dostupné nástroje PhoneGap Desktop Application a PhoneGap CLI. Jde o podobné nástroje určené pro správu projektů a jejich testování. Desktop Application pro práci nabízí jednoduché grafické uživatelské rozhraní, zatímco CLI nabízí rozhraní příkazové řádky. K testování na mobilním zařízení je určena aplikace PhoneGap Mobile Developer. Zatímco Desktop Application a CLI vytváří a upravují PhoneGap projekt, Mobile Developer umožňuje aplikaci předvést a otestovat a nepotřebuje k tomu žádné další SDK[2] pro specifickou platformu. Mobile Developer představuje sandbox s přednastavenými zásuvnými moduly. Díky tomu umožňuje rychlou a účinnou cestou otestovat aplikace.

PhoneGap Build je cloudové úložiště pro PhoneGap, kde je možné nahrát a spravovat. PhoneGap Build projekt sám převede na hybridní aplikaci pro každou podporovanou platformu. Výslednou aplikaci je možné okamžitě stáhnout jak do počítače, tak i do mobilního zařízení. Protože výstupem PhoneGap Build je již hybridní aplikace, není potřeba využívat k jejímu testování mobilní aplikaci PhoneGapu.

2.4 Vývoj aplikace ve PhoneGap

V této fázi vývoje se bude aplikace vytvářet pouze lokálně na počítači. Je výhodnější projekt spravovat pomocí PhoneGap CLI, protože oproti desktopové aplikaci nabízí více funkcí. Jde zejména o příkazy pro závěrečnou část vývoje, například příkaz pro kompilaci projektu v hybridní aplikaci pro vybranou platformu. Pro testovací účely na mobilním zařízení bude sloužit aplikace PhoneGap Mobile Developer. Jedná se o nejjednodušší možnost testování aplikace, protože kromě CLI a Mobile Developeru není potřeba již nic dalšího instalovat.

Workflow lokálního vývoje aplikace je následující:

1. Nainstalování PhoneGap CLI a PhoneGap Mobile Developer
2. Vytvoření PhoneGap projektu
3. Nastavení projektu
4. Lokální testování aplikace

2.4.1 Instalace PhoneGap CLI

Před instalací PhoneGap CLI je nezbytné nejdříve nainstalovat `Node.js` a `git`.

- `Node.js` - jedná se o JavaScript runtime prostředí pro vývoj serverových a síťových aplikací. PhoneGap CLI vyžaduje několik knihoven od Node.js pro různé příkazy.

[2] Jedná se o sadu vývojových nástrojů pro vytváření specifických aplikací

- `git` - představuje distribuovaný řídící systém používaný pro správu zdrojových kódů. PhoneGap CLI využívá git na pozadí pro stahování různých souborů, například pro stažení šablony projektu.

Instalace PhoneGap CLI se provádí přes příkazovou řádku (pro systémy Windows) nebo terminál (pro MAC) napsáním npm příkazu

```
npm install -g phonegap@latest
```

Tento příkaz vytvoří složku „`node_modules`“ obsahující vše potřebné ke spuštění PhoneGap CLI.

Vlajka `-g` znamená, že lokace vytvářené složky má být globální, aby se k ní dalo přistupovat na daném zařízení odkudkoliv. Například v operačním systému MAC se jedná o lokaci „`\usr\local\lib\node_modules\phonegap`“. Otestovat, že je program nainstalován správně lze pomocí příkazu „phonegap“. Pokud vše bylo úspěšné, měl by se v příkazové řádce objevit podobný text jako v obrázku 3.

```
C:\>phonegap

Usage: phonegap [options] [commands]

Description:

  PhoneGap command-line tool.

Commands:

  help [command]       output usage information
  create <path>        create a phonegap project
  build <platforms>    build the project for a specific platform
  install <platforms>  install the project on for a specific platform
```

Obrázek 3: Výpis při napsání příkazu „phonegap“

2.4.2 Instalace PhoneGap Mobile Developer

Protože se jedná o mobilní aplikaci, její instalace se provádí přes obchody s aplikacemi. Aplikace se v těchto obchodech nachází pod názvem „PhoneGap Developer.“ Momentálně je možné aplikaci stáhnout z iTunes, GooglePlay nebo Windows Phone Store. Protože každý obchod poskytuje aplikaci pouze pro svou platformu, je možné tímto způsobem aplikace odzkoušet pouze na zařízeních platformy iOS, Android a nebo Windows Phone.

2.4.3 PhoneGap projekt

PhoneGap projekt je strukturovaná soustava složek určená pro zdrojové soubory projektu, přídavné zásuvné moduly, dodatečné kódy a také výsledné hybridní aplikace.

Nový projekt se vytváří příkazem

```
phonegap create <path> [options]
```

Parametr `<path>` je povinný parametr příkazu a určuje lokaci vytvářeného projektu. Je možné zadat i další parametry v místě `[options]`, jmenovitě:

- `--id` - unikátní identifikátor projektu napříč obchody s aplikacemi. Doporučuje se používat opačně formátovaný jmenný prostor domény, např. „com.phonegap.helloworld.“
- `--name` - zobrazovaný název aplikace. Společně s ID se nastaví do manifestu aplikace (soubor config.xml) a používají se při vytváření nativního projektu.
- `--copy-from`, `--src <path>` - vytvoří projekt využitím kopie již existujícího projektu specifikovaného zadanou cestou `<path>`.
- `--link-to <path>` - jedná se o symlink[3]/zkratku do adresáře www aniž by se kopíroval.
- `--template <npm package | path | git url>` - vytvoří aplikaci využitím šablony. Je možné využít šablonu publikovanou na npm, lokální šablonu v zadaném umístění nebo šablonu nacházející se na daném git URL.
- `config` - toto nastavení umožňuje předat JSON[4] string s konfiguračními parametry, na kterých některé pluginy závisí.

Každý projekt musí mít nastavené jméno (`name`) a identifikátor (`id`). V případě, že při vytváření projektu nebyly tyto parametry zadané, budou PhoneGapem implicitně doplněny ze šablony „hello world.“ Příkazy mohou vypadat jako v ukázce níže.

```
phonegap create path/to/my-app
phonegap create path/to/my-app "com.dashboard.app" "Dashboard App"
phonegap create path/to/my-app --id "com.dashboard.app" --name "Dashboard App"
phonegap create path/to/my-app --copy-from ../CLI_App
```

Protože projekt neslouží pouze pro zdrojové kódy, má každý projekt předem danou strukturu. Na nejvyšší úrovni projektu se nachází:

- .cordova - tato složka se implicitně vytváří, pokud je projekt vytvořen přes desktopovou aplikaci. U CLI se vytvoří pouze pokud je předán i parametr `config`.
- hooks - implicitně prázdná složka. Představuje místo pro vlastní skripty, které mohou být přidány vývojáři aplikace nebo zásuvných modulů a nebo vlastním systémem pro upravení cordova příkazů. Ve složce hooks se vytvoří podsložka s názvem skriptu a do ní se vloží skript. Nicméně je nutné podotknout, že se jedná o zastaralý způsob. Doporučený styl používání hooks je přes `<hook>` element přímo v XML souboru (například v souboru config.xml). I přesto, že se jedná o zastaralý způsob, se při kompilaci vždy jako první kontroluje právě složka hooks, a až poté se kontrolují jednotlivé XML soubory.

[3] Jde o soubor odkazující na jiný soubor
[4] JSON představuje syntaxi pro ukládání a výměnu dat a alternativu k XML

- platforms - do této složky PhoneGap postaví hybridní aplikaci pro jednotlivé platformy.
- plugins - v této složce se nachází všechny pluginy projektu
- www - tato složka obsahuje webovou část aplikace.
- config.xml - XML soubor obsahující parametry aplikace jako její název, ID, jméno autora apod.

Ve složce www by se měl na nejvyšší úrovni nacházet soubor index.html a spec.html. Implicitně jsou zde vytvořeny i složky css, img, js, spec a v některých případech i složka res.

Soubor index.html představuje úvodní stránku aplikace, ale také kořen celé webové části projektu. Z tohoto důvodu musí být vždy v nejvyšší úrovni v projektu. Žádný jiný soubor webové části nesmí být ve vyšší úrovni.

Soubor spec.html a složka spec slouží pro testování javascriptu prostřednictvím aplikačního rámce Jasmine. Jsou zde pouze pro výpomoc vývojářům a projekt samotný na nich není závislý.

Ostatní složky nejsou povinné. I když žádný soubor webové části nesmí být na vyšší úrovni než index.html, na stejné úrovni být může. Zbylé složky proto slouží pouze k rozdělení ostatních souborů pro přehlednost.

Projekt obsahuje také speciální soubor nazývaný „.pgbomit“ (standardně ve složce res). Tento soubor značí, že PhoneGap Build nemá zahrnout obsah této složky do balíčku aplikace. Funkce takové složky je v tom, že může obsahovat soubory vyžadované PhoneGapem až po kompilaci. Typickým příkladem použití je složka obsahující ikony pro různé platformy. Pokud je do této složky přidán soubor .pgbomit, tak její obsah nebude přidán do balíčku aplikace, kromě ikon, které byly použity pro danou platformu. Tímto přístupem je možné výrazně snížit velikost výsledné aplikace u projektu, který je zamýšlen pro mnoho platforem.

Důležitým souborem je config.xml. Jedná se o soubor, který umožňuje vývojářům snadno specifikovat metadata o jejich aplikaci. Pro úspěšné načtení tohoto souboru je potřeba, aby byl na stejné úrovni jako index.html. Nejdůležitějšími prvky jsou:

- `<widget>` - tento element představuje kořen souboru. Je potřeba v něm nastavit následující atributy:
  - `id` - unikátní identifikátor projektu napříč obchody s aplikacemi. Doporučuje se používat opačně formátovaný jmenný prostor domény, např. „com.phonegap.helloworld.“
  - `version` - verze aplikace. Doporučení je používat číselné označení jako například „0.0.1“.
  - `versionCode` - nepovinný atribut. Při vyvíjení aplikace pro Android se může pomocí tohoto atributu nastavit versionCode pro zajištění kompatibility.
- `<name>` - specifikuje název aplikace,
- `<description>` - pro popis aplikace,

- `<platform>` - Určuje platformu, pro kterou se má aplikace postavit. Config.xml může obsahovat i více těchto elementů, pokud má postavit aplikaci pro více platform. Element obsahuje atribut name, do kterého se vepisuje název platformy. Podporované hodnoty atributu jsou 'android', 'ios', 'winphone'. Například `<platform name="ios" />`.
- `<plugin>` - Slouží pro přidání pluginu psaného v nativním kódu.

2.4.3.1 Zásuvné moduly

Důležitou součástí každé aplikace jsou tzv. zásuvné moduly. „Zásuvný modul je softwarový doplněk instalovaný na program, a tím mu umožní vykonávat další funkce. Jako příklad, Internetový prohlížeč umožňuje uživatelům nainstalovat pluginy do prohlížeče a tím mu umožnit vykonávat funkce, které v základní instalaci nejsou.“[5]

Mobilní aplikace již v základu obsahuje velké množství předem nainstalovaných zásuvných modulů. Ve většině případů se jedná o zásuvné moduly, které umožní hybridní aplikaci využívat hardwarové a softwarové funkce zařízení. Celý seznam zásuvných modulů, které je možné použít ve PhoneGap projektu, je rozsáhlý a stále se rozšiřuje. Může se jednat například o tyto pluginy:

- Device Orientation, který využívá u zařízení funkci compass. Compass je senzor, který zjišťuje směr, kterým je zařízení natočeno.
- Dialogs, který umožňuje použití nativních dialogových UI elementů.
- Camera, který definuje globální camera objekt poskytující API[5] pro focení a vybírání obrázků ze systémového alba.

PhoneGap podporuje dva typy zásuvných modulů. Jedná se o moduly psané v nativním jazyce a moduly psané v javascriptu.

Moduly psané v nativním jazyce se přidávají do projektu nastavením souboru config.xml. Na tyto moduly se odkazuje využitím elementu `<plugin>`. V elementu `<plugin>` se nastavuje:

- `name` - atribut. Na zásuvné moduly by měl odkazovat jejich identifikátor. Identifikátor bývá psán jako opačně formátovaný jmenný prostor. Atribut je nepovinný, pokud je plugin zálohován na gitu.
- `spec` - nepovinný, ale velmi doporučovaný atribut. Specifikuje jaká verze modulu má být použita. Pokud je tento atribut vynechán, bude vždy stažena nejnovější verze - to může vést k neočekávanému chování aplikace. Tento atribut také může označovat git repositář u modulů zálohovaných na gitu. Pokud atribut obsahuje úplné URL, předpokládá se, že se jedná o modul, který je uložený na gitu. Tento atribut může obsahovat i operátor „∼“ (tilda), který zajišťuje stáhnutí nejnovější verze pluginu se stejnou hlavní verzí.

[5] Jedná se o rozhraní pro programování aplikací

- `source` - nepovinný atribut. Představuje zdroj pluginu a může se jednat o pgb, npm nebo git. Implicitně se používá npm (nebo git pokud je nalezeno URL).
- `<param>` - element. Některé pluginy vyžadují přednastavené konfigurační informace a tyto konfigurace se provádí právě pomocí tohoto elementu.

U ukázek ve výpisu 1 bude první plugin stažen z git repositáře. Druhý plugin bude stažen z pgb jako nejnovější „1.x.y“ verze (dané tildou před číslem verze). Mohlo by jít například o verzi 1.9.2, i když nejnovější verze pluginu by mohla být 3.0.2.

```
<plugin spec="https://github.com/apache/cordova-plugin-file.git#4.1.0" source="
    git" />

<plugin name="example-plugin" source="pgb" spec="~1">
   <param name="APIKey" value="12345678" />
   <param name="APISecret" value="12345678" />
</plugin>
```

Výpis 1: Ukázky XML pluginů

Javascriptové moduly se přidávají tak, že se na daný modul odkáže v souboru index.html. K odkazování se používá prvek `<script>`. V ukázce ve výpisu 2 se načítají pluginy PhoneGap API a skener čárových kódů.

Na některé moduly není potřeba odkazovat pomocí elementu `<script>`, pokud modul využívá element `js-module`. Pomocí tohoto elementu Cordova automaticky stáhne a nastaví JavaScript soubory daného modulu, není tedy potřeba odkazovat na modul pomocí elementu `<script>`. Tímto stylem se obvykle nastavují hlavní Cordova moduly. U modulů třetích stran závisí na tom, jak jsou implementovány, a proto je potřeba u nich zjistit z dokumentace, zdali je nebo není potřeba na ně odkazovat explicitně.

```
<script src="cordova.js"></script>
<script src="barcodescanner.js"></script>
```

Výpis 2: Ukázky JavaScriptových pluginů

2.4.4 Nastavení projektu

Tato část se věnuje dodatečnému nastavení projektu. Toto nastavení je důležité především z toho důvodu, že webové stránky implicitně nejsou přizpůsobené práci na mobilním zařízení. Následující nastavení umožňují správné načtení webové části projektu a PhoneGap API.

Soubor index.html musí v hlavičce obsahovat následující nastavení:

```
<meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale
    =1, minimum-scale=1, width=device-width, height=device-height, target-
    densitydpi=device-dpi" />
```

Výpis 3: Nastavení stránky pro mobilní zařízení

Meta tag ve výpisu 3 nastavuje stránku pro mobilní zařízení. Nastavuje velikost obsahu na 100% (initial-scale), velikost zařízení na maximální šířku a výšku a znemožňuje přenastavit velikost (user-scalable=no).

```
<script type="text/javascript" src="cordova.js"></script>
```

Výpis 4: Odkaz na PhoneGap API

Skript ve výpisu 4 odkazuje na neexistující soubor cordova.js. Tento soubor je generován až při kompilaci aplikace. Projekt by neměl před kompilací tento soubor obsahovat, protože každá platforma používá jinou verzi cordova.js. Pokud by byla použita nesprávná verze tohoto souboru, může dojít ke zhroucení aplikace. Ekvivalentem cordova.js je soubor phonegap.js. Tento soubor je při kompilaci nahrazen souborem cordova.js, stahují se tedy na něj stejná pravidla.

Soubor index.html používá soubor index.js. Tento soubor není v projektu nutný, ale nově vytvořeným projektům přidává malou aplikační logiku pro zjištění, kdy byla knihovna Cordova načtena a připravena k použití. Index.js je používán v těle index.html pomocí skriptu ve výpisu 5.

```
<script type="text/javascript">
    app.initialize();
</script>
```

Výpis 5: Inicializace po spuštění

3 Architektura aplikace

Tato část práce se věnuje architektuře vyvíjené aplikace. První část této kapitoly se zabývá volbou architektury aplikace z pohledu komunikace po internetové síti. Druhá část se zabývá výběrem architektury pro grafické uživatelské rozhraní.

„Softwarová architektura programu nebo výpočetního systému je struktura nebo struktury systému, které zahrnují softwarové prvky, externě viditelné vlastnosti těchto prvků a vztahy mezi nimi.“[4]

Jedním z prvních úkolů při návrhu mobilní aplikace je zvolení architektury software. Správně či špatně zvolená architektura v konečném důsledku ovlivní kvalitu aplikace, její údržbu, výkonnost a celkový úspěch.

3.1 Architektura klient-server

V nejobecnějším měřítku se bude jednat o architekturu klient-server. Jedná se o architekturu, která rozděluje pracovní zátěž mezi poskytovatele služby nebo zdroje a žadatele služby. Poskytovatel se nazývá server a žadatel klient. V případě této práce bude serverem Google Analytics, který bude poskytovat údaje o stránkách katedry informatiky. Klientem bude dashboardová aplikace.

Nyní je potřeba vybrat architekturu klienta. Výběr architektury určí, co vše bude aplikace na zařízení podporovat a jak moc závislá bude na serveru. Nejznámějšími architekturami jsou:

- fat-client (tlustý klient),
- thin-client (tenký klient),
- smart-client (chytrý klient).

3.1.1 Tlustý a tenký klient

Aplikace stavěné jako tlustý klient využívají lokální hardwarové zdroje a funkce platformy operačního systému, na kterém běží. Takové aplikace provádí největší množství operací a zpracování. U tlustého klienta se nevyžaduje neustálé připojení k internetu, klient většinou komunikuje s databází za účelem poslání archivních dat. Tlustý klient je schopen pracovat v offline režimu.

Nevýhodou tlustého klienta je jeho náročnost na nasazení a údržbu. Čím více se složitost aplikace a klientské platformy zvyšuje, o to složitější je nasazení aplikace na klientské zařízení spolehlivou a bezpečnou cestou.

Architektura tenkého klienta byla vytvořena jako řešení problémů s nasazením a údržbou aplikace. Jedná se o aplikace využívající webový prohlížeč, kde nasazení a aktualizace probíhá na centrálním webovém serveru. Díky tomu odstraňují potřebu explicitně nasazovat a spravovat aplikaci či její část na zařízení klienta. Tím umožní aplikaci nasadit rychle a efektivně pro široké

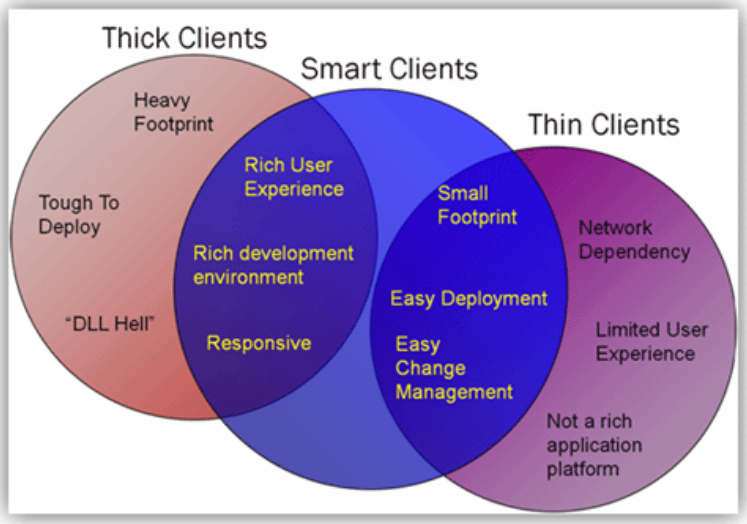
spektrum uživatelů. Největší množství operací provádí server, klient pouze zobrazuje příchozí data a na server zasílá požadavky. Klient představuje jednoduchý terminál k serveru.

Problémem tenkého klienta je, že vyžaduje neustálé připojení k síti, protože vše provádí server. Tenký klient není schopen vůbec pracovat v offline režimu. Také nenabízí k dispozici obvyklé funkce zařízení jako je drag-and-drop, „zpět“ a „znovu“ a jiné. Tím se snižuje použitelnost aplikace. Tenký klient také trpí sníženou výkonností, pokud po síti přenáší velká data (například multimédia) nebo pokud má k dispozici pomalé připojení k internetu.

Z textu by mělo být zřejmé, že pro dashboardovou aplikaci se tenký a tlustý klient nehodí. Tlustý klient se nehodí kvůli tomu, že cílem je vytvořit multiplatformní aplikaci, tlustý klient ale trpí problémy s nasazením a údržbou. Tenký klient se nehodí kvůli tomu, že je přímo závislý na připojení k internetu. Neustálé připojení a přenášení dat představuje pro mobilní zařízení problém. Zbývajícím řešením je tedy postavit aplikaci na architektuře chytrého klienta.

3.1.2 Chytrý klient

Chytrý klient představuje „zlatou střední cestu“ mezi tlustým a tenkým klientem. Tento klient není tak jasně definovaný jako předchozí dva klienti a nemusí plně využívat všech předností, naopak jeho povaha často závisí na designu a implementaci konkrétní aplikace. Cílem chytrého klienta je využít předností tlustého a tenkého klienta. U tlustého klienta se jedná především o využití lokálních zdrojů a obsloužení dočasně připojených uživatelů, u tenkého klienta jde přednostně o využití síťových zdrojů, inteligentní instalaci a aktualizace a flexibilitu. Obrázek 4 přednosti jednotlivých klientů shrnuje.



Obrázek 4: Shrnutí tlustého, tenkého a chytrého klienta
Zdroj: https://i-msdn.sec.s-msft.com/dynimg/IC139030.gif

Pro dashboardovou aplikaci je nejvhodnější právě chytrý klient, protože v mnoha směrech jsou jeho přednosti přímo požadované hybridními aplikacemi. Například díky tomu, že využívá lokální zdroje, může dashboardová aplikace nabídnout uživateli jak bohaté a responzivní uživatelské rozhraní, tak také efektivní možnosti zpracovávání dat. Aplikace může využít i lokálních hardwarových funkcí, ke kterým přistupuje pomocí zásuvných modulů.

Dashboardové aplikaci vyhovuje podpora dočasně připojených uživatelů, protože nevyžaduje neustálé připojení k internetu, naopak data pouze v periodických intervalech aktualizuje. Uživatelům umožní pracovat i v offline režimu tím, že data pouze přestane aktualizovat.

Cílem chytrého klienta je také nabízet inteligentní instalaci a aktualizaci aplikace. Toho se může snadno docílit umístěním aplikace do obchodu s aplikacemi.

3.2 Struktura uživatelského rozhraní

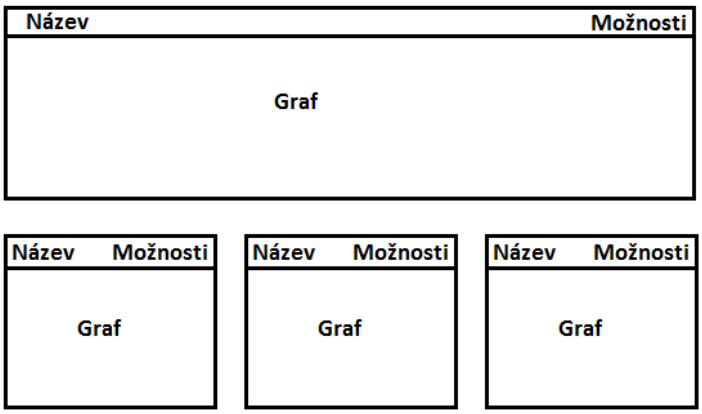
Důležitou částí aplikace je grafické uživatelské rozhraní. Grafické uživatelské rozhraní (zkráceně GUI) je definováno jako „software, který pracuje v místě kontaktu (rozhraní) mezi počítačem a uživatelem, a které využívá grafických prvků (dialogová okna, ikony, menu, posuvníky) místo textových znaků, aby uživatel mohl zadávat příkazy počítači nebo manipulovat s tím, co je na obrazovce.“[8]

Pro dashboardovou aplikaci je výběr architektury grafického rozhraní důležitý, protože má graficky zobrazovat data a tato data musí být prezentována ve snadno pochopitelné formě. Zvolení špatné architektury GUI může vést k situacím, kdy grafické prvky odpoutávají pozornost uživatele nebo stěžují práci s aplikací.

Aplikace bude využívat tzv. responzivní design. Responzivní design znamená, že se jednotlivé komponenty stránky aplikace přizpůsobují velikosti aplikačního okna. Pokud je příliš úzké, posunou se některé komponenty níže, než aby se všechny zmenšily do těžko rozlišitelných rozměrů. Responzivní design je důležitý právě pro mobilní zařízení, protože mívají nižší rozlišení než počítače.

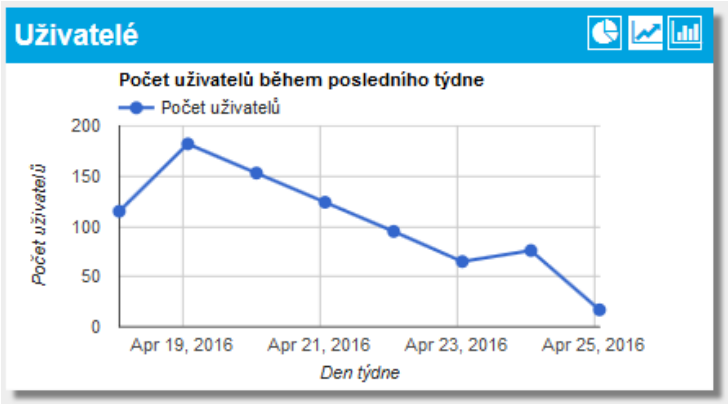
Aplikace se bude skládat ze čtyř částí, kde každá část představuje samostatnou komponentu pro zobrazení určitého typu dat. Nejdůležitější z těchto komponent je komponenta ukazující graf návštěvnosti. Z tohoto důvodu je komponenta umístěna hned nahoře a rozprostírá se po celé šířce aplikace. Zbylé tři komponenty ukazují historická data. Protože se tato data tolik nemění, jsou tyto komponenty menších rozměrů a umístěny vedle sebe. Skica v obrázku 5 ukazuje rozmístění.

Grafické rozhraní bude postaveno na základě tzv. Flat designu. Flat design je minimalistické uživatelské rozhraní, které klade důraz na minimální užití stylistických prvků (jako jsou stíny, zaoblení, textury či gradienty a ostatní elementy vytvářející 3D vzhled) nebo jejich úplné vynechání a místo toho se zaměřuje na užití jednoduchých prvků, typografie a jednotných barev. Typickým příkladem uživatelského rozhraní využívající Flat design je rozhraní Windows 8. Výhodou tohoto rozhraní je, že grafické prvky zbytečně neodvádí pozornost, snadněji sdělují



Obrázek 5: Skica pro rozvržení aplikace

informace a ulehčují návrh responzivního rozhraní. Obrázek 6 zobrazuje, jak vypadá jeden z modulů pro zobrazení grafu při použití flat designu.



Obrázek 6: Modul zobrazující údaje o počtu uživatelů s použitím Flat designu

4 Zpracování dat

Tato část se zaobírá problematikou zpracování dat. Vysvětluje, jak se vytváří dotazy na server a v jaké formě server posílá zpět odpověď.

Data poskytují servery od Google Analytics. K datům se přistupuje dvěma způsoby.

1. Pomocí Google Analytics API (zkráceně „ga API“) metodou

   ```
   analytics.data.ga.get()
   ```

2. Zásláním RESTful GET požadavku.

Metoda `analytics.data.ga.get()` přijímá jako parametr javascriptový objekt obsahující parametry dotazu. Dotaz může vypadat například jako ve výpise 6. Tento dotaz vrátí všechny počet uživatelů za posledních sedm dní.

```
var queryOptions = {
   'ids': 'ga:12345'
    'metrics': 'ga:users',
    'start-date': '7daysAgo',
    'end-date': 'today'
};
analytics.data.ga.get(queryOptions);
```

Výpis 6: Dotaz pomocí metody ga.get()

RESTful požadavek může podobat požadavku ve výpisu 7. V této ukázce ve výpisu 7 je dotaz od uživatele s id 12345, který vyhledává celkový počet relací (sessions) a počet jednostránkových relací (bounces) v období mezi 1.10. 2008 až 31.10. 2008.

```
Authorization: Bearer {oauth2-token}

GET https://www.googleapis.com/analytics/v3/data/ga
  ?ids=ga:12345
  &start-date=2008-10-01
  &end-date=2008-10-31
  &metrics=ga:sessions,ga:bounces
```

Výpis 7: RESTful dotaz

Ať už se jedná o dotaz pomocí metody z ga API nebo jako RESTful GET dotaz, musí obsahovat čtyři základní parametry - `ids`, `metrics`, `start-date` a `end-date`. Důvod je zřejmý - každý dotaz musí obsahovat kdo (`ids`) žádá data, co se žádá (`metrics`) a časové rozmezí od-do. Existují i další parametry, jako je například `dimensions`, které určuje, jak se mají příchozí data

rozdělit (například podle dne v týdnu), nebo `filter`, který nastavuje filtrování pro příchozí data.

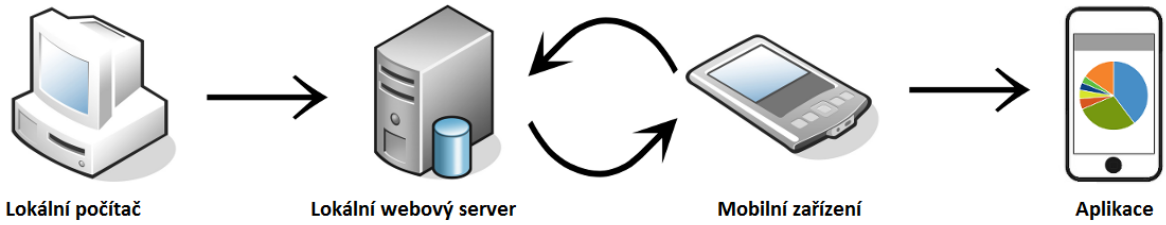
Server odesílá odpověď v podobě JSON struktury nebo jako Data Table objekt. Google Analytics implicitně posílá odpovědi jako JSON strukturu. Struktura obsahuje mimo jiné tyto prvky:

- `id`, který obsahuje URL dotazu,
- `query`, který obsahuje objektově zapsaný dotaz,
- `profileInfo`, kde se nachází informace o profilovém účtu,
- `columnHeaders`, jehož hodnotou je pole objektů obsahující údaje k jednotlivým sloupcům,
- `rows`, ve kterém se nachází jednotlivé záznamy uložené po řádcích.

Pokud se v dotazu nastaví parametr `output` na `"dataTable"`, bude server posílat zpět Data Table objekt. Odpověď bude stále posílána zpět jako JSON struktura, ale místo prvku `rows` bude obsahovat prvek `dataTable`, který obsahuje objekt s prvky `cols` a `rows` představující sloupce a řádky tabulky. Právě odpovědi ve formátu Data Table bude dashboardová aplikace používat, protože k zobrazování grafu využívá Google Charts a Data Table je strukturován tak, aby mohl být okamžitě použit jako parametr pro vykreslení grafu.

5 Testování

PhoneGap umožňuje vyvíjenou aplikaci odzkoušet, aniž by bylo potřeba projekt předem kompilovat pro vybranou platformu. Přes PhoneGap CLI se projekt nejdříve aktivuje a poté se spustí na mobilním zařízení pomocí PhoneGap Mobile Developer. Jak je ilustrováno v obrázku 7, při aktivaci je projekt nahrán na lokální webový server. Mobile Developer po připojení projekt stáhne včetně potřebných souborů pro spuštění projektu na mobilním zařízení dané platformy.



Obrázek 7: Interakce mezi PhoneGap CLI a Mobile Developer

Ke spuštění lokálního webového serveru a nahrání projektu slouží příkaz

```
phonegap serve [options]
```

Parametr `[options]` je nepovinný, slouží k dodatečnému nastavení. Lze použít toto nastavení:

- `--port`, `-p <n>` - nastaví port pro webový server. Implicitně je nastaven port 3000.
- `--autoreload` - obnoví aplikaci při změně zdrojových souborů. Implicitně je tato možnost aktivována.
- `--no-autoreload` - zakazuje autoreload.
- `--browser` - zapíná podporu pro desktopové prohlížeče. Implicitně aktivováno.
- `--no-browser` - vypíná podporu prohlížečů.
- `--localtunnel` - zapíná lokální tunel pro veřejný přístup. Implicitně vypnuto.

Příkaz pro spuštění serveru může vypadat jako jeden z následujících příkazů:

```
phonegap serve
phonegap serve --port 1337
phonegap serve --no-browser
```

[phonegap documentation] Pokud je vše správně, po chvíli se spustí webový server a v CLI se vypíše jeho IP adresa. Nyní, když je server přichystán, se stačí k němu připojit pomocí PhoneGap Mobile Developer. Po spuštění Mobile Developeru se zobrazí okno pro připojení k serveru. Do pole se zadává IP adresa lokálního webového serveru.

Po zadání IP adresy a připojení k serveru se zobrazí informační zprávy `downloaded` (staženo) a `extracted` (extrahováno). Pokud vše dopadlo dobře, napíše se zpráva `success` (úspěch) a v zařízení se načte PhoneGap projekt. Pokud došlo k chybě, napíše se zpráva `error` (chyba) a v takovém případě je potřeba zkontrolovat síťové připojení zařízení a lokálního webového serveru a nastavení projektu.

Pro připojení k serveru je potřeba, aby byly webový server a mobilní zařízení připojeni ke stejné počítačové síti a aby byl PhoneGap projekt na webový server nahrán. Je možné provádět jakékoliv změny v projektu i když je projekt aktivován. Jakékoliv změny se po uložení okamžitě promítnou v připojené mobilní aplikaci. Server se vypíná stisknutím kláves CTRL a C a potvrzením vypnutí.

5.1 Debugování

Jednou z důležitých částí vývoje aplikace je debugování. „Debugování je proces lokalizování a opravování nebo obcházení bugů (chyb) v počítačovém programovém kódu.“[6]
Existuje mnoho způsobů jak debugovat aplikaci. PhoneGap primárně pro debugování využívá nástroj weinre. Weinre je zkratka pro **WE**b **IN**spector **RE**mote a jedná se o nástroj pro debugování zařízení. Weinre nabízí debugování lokálně sledováním lokálního webového serveru s nahraným PhoneGap projektem. Je možné debugovat také vzdáleně pomocí PhoneGap Build.

U lokálního debugování se sleduje komunikace mezi lokálním webový serverem a aplikací v Mobile Developeru. Weinre není součástí instalace PhoneGap, je potřeba ho dodatečně nainstalovat. Před instalací je potřeba mít nainstalovaný Node.js. Weinre se instaluje přes rozhraní příkazové řádky následujícím příkazem.

```
npm install -g weinre
```

Weinre se spouští následujícím příkazem

```
weinre --boundHost A.B.C.D
```

Za „A.B.C.D“ je potřeba vepsat lokální IP adresu počítače, na kterém běží server. Pokud je již přes PhoneGap CLI spuštěný lokální webový server, je možné lokální IP adresu vyčíst právě z PhoneGap CLI. Příkaz může vypadat například následovně:

```
weinre --boundhost 192.168.1.20
```

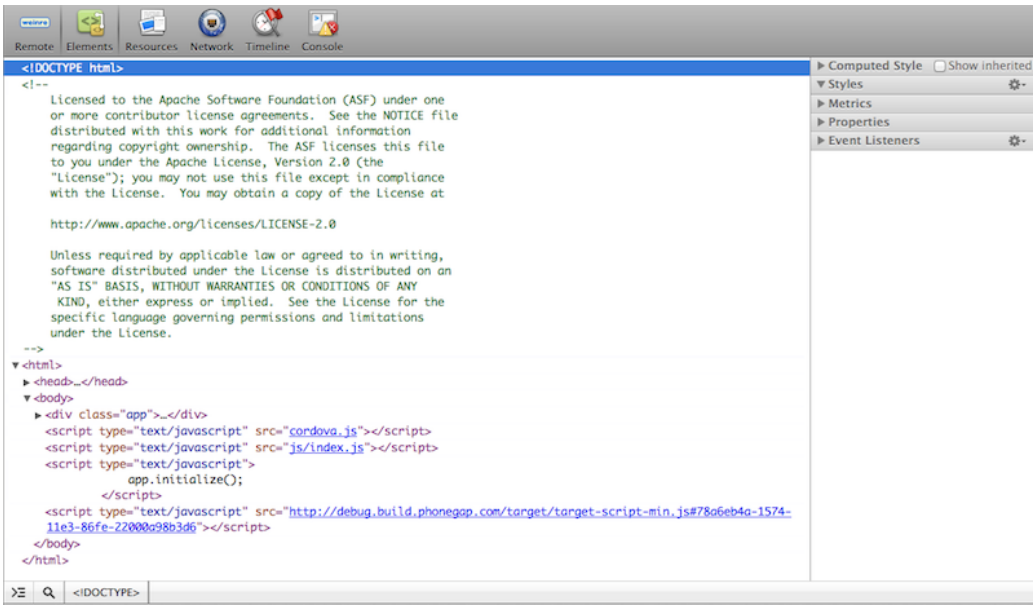
Dále je potřeba propojit weinre s mobilní aplikací. Toho se docílí přidáním `<script>` elementu, který odkazuje na weinre, do hlavičky souboru index.html. Element by měl být nastaven podobně jako ve výpisu 8. IP adresa ve výpisu 8 by měla být shodná s IP adresou, pod kterou je weinre spuštěný.

```
<script src="http://A.B.C.D:8080/target/target-script-min.js#anonymous">
</script>
```

Výpis 8: Nastavení index.html pro debugování

K rozhraní weinre se přistupuje přes webový prohlížeč a připojení se na URL, pod kterým je spuštěn weinre. Při spuštění aplikace v Mobile Developeru bude weinre spárován s aplikací běžící na lokálním serveru. V prohlížeči se zobrazí několik odkazů. Je potřeba najet na odkaz s názvem „debug client user interface“ (debugování uživatelského rozhraní klienta) a poté na další stránce kliknout na odkaz v sekci „Targets“. Po kliknutí se odkaz obarví na zeleno, tím je oznámeno úspěšné propojení aplikaci s weinre.

Weinre umožňuje aplikaci debugovat podobně jako se debugují webový stránky pomocí nástrojů pro vývojáře v samotném webovém prohlížeči. Například elementy, na které se kurzorem ukáže v debugovém režimu, budou zvýrazněny v připojené aplikaci. Sloupec v pravé části obrazovky zobrazuje nastavení kaskádových stylů. Je podporována javascriptová konzole a dále i záložky Network, Timeline a Resources zobrazující datový přenos po síti. Obrázek 8 zobrazuje rozhraní weinre.



Obrázek 8: Webové rozhraní weinre
Zdroj: http://docs.build.phonegap.com

6 Možnosti nasazení

Po tom, co byl PhoneGap projekt nastaven a otestován, se vývoj aplikace dostává do fáze nasazení. Tato část pokrývá problematiku vytvoření výsledné aplikace z PhoneGap projektu. V tomto případě jde o proces zabalení webové části projektu do nativního kódu vybrané platformy a tím vytvoření hybridní aplikace. Předělat PhoneGap projekt v hotovou hybridní aplikaci je možné

- využitím SDK pro danou platformu,
- nahráním projektu na cloudové uložiště PhoneGap Build,
- vzdáleným připojením k PhoneGap Build přes PhoneGap CLI,
- lokálně pomocí PhoneGap CLI.

Při stavění aplikace pomocí SDK pro danou platformu je PhoneGap použit pouze jako doplněk. Tento způsob je časově nejnáročnější, protože vyžaduje instalaci, znalost a správné nastavení každého prostředí, ve kterém se aplikace kompiluje. Například pro vytvoření multiplatformní aplikace zaměřené na operační systémy Android a iOS, je potřeba nainstalovat a poté aplikaci zkompilovat v Android SDK pro Android a Xcode pro iOS. Výhodou tohoto přístupu je možnost pokrýt velké množství platforem.

Nejjednodušší, ale zároveň nejomezenější možností je kompilace aplikace přes PhoneGap Build. PhoneGap Build celý proces automatizuje, takže po nahrání projektu bude projekt okamžitě zkompilován v aplikace pro všechny podporované platformy bez nutnosti dalších nastavení. Nevýhodou ovšem je, že PhoneGap Build podporuje pouze operační systémy Android, iOS a Windows Phone.

Alternativou je také možnost využít PhoneGap CLI pro vzdálené připojení k PhoneGap Build. PhoneGap CLI zde funguje jako médium pro vzdálené připojení a PhoneGap Build se stará o kompilaci. Protože se kompilace provádí vzdáleně, odpadá potřeba jakýchkoliv SDK na lokálním počítači. Nicméně na výběr jsou opět pouze platformy Android, iOS a Windows Phone. K PhoneGap Build se vzdáleně připojuje přes příkaz

```
phonegap remote [command]
```

Parametrem `[command]` může být jedno z následujících:

- `login` - přihlášení k PhoneGap Build
- `logout` - odhlášení z PhoneGap Build
- `run <platform>` - postaví a nainstaluje aplikaci pro zadanou platformu.

Pokud je použitý parametr `login`, bude uživatel vyzván k zadání přihlašovacího loginu a hesla ke cloudové službě. K přihlášení je možné použít také AdobeId. Po použití parametru `run` bude projekt nahrán na PhoneGap Build a po kompilaci bude v terminálu vygenerován QR kód. Tento kód může být mobilním zařízením oskenován a použit pro stáhnutí aplikace. Typickým příkladem použití může být následující sekvence příkazů:

```
phonegap remote login
phonegap remote run ios
phonegap remote logout
```

Poslední možností je lokální kompilace aplikace pomocí PhoneGap CLI. Jedná se o nejefektivnější možnost, pokud je cílem pokrýt co nejvíce platforem. PhoneGap CLI využívá pouze jediný příkaz pro kompilaci aplikace pro všechny podporované platformy a využitím Cordova příkazů umožňuje aplikace vytvářet i pro jiné platformy, než jsou Android, iOS nebo Windows Phone. Samotný PhoneGap CLI není schopen projekt předělat v aplikaci, místo toho využívá nainstalované SDK pro jednotlivé platformy. I když je potřeba mít nainstalované různá SDK, PhoneGap práci ulehčuje právě tím, že sám využije SDK k zabalení aplikace.

Po nainstalování potřebných SDK se aplikace kompiluje následujícím příkazem:

```
phonegap run [<platforms>] [options]
```

Tento příkaz aplikaci sestaví a také ihned nainstaluje. Po kompilaci bude v projektu ve složce „Platforms“ již přichystaná aplikace. Například pro platformu Android bude ve složce „../Platforms/Android.“

Parametr `[<platforms>]` určuje platformy, pro které bude aplikace vytvořena. Pokud je tento parametr vynechán, aplikace bude vytvořena pro všechny podporované platformy. Implicitně bude aplikace nainstalována na připojené zařízení. Pokud žádné nebude nalezeno, bude nainstalována na emulátor. V možnostech `[options]` můžou být tyto parametry:

- `--device` - aplikace bude nainstalována pouze na zařízení,
- `--emulator`, `-e` - aplikace bude nainstalována pouze na emulátor,
- `--target <id>` - nainstaluje aplikace na zadaný cíl,
- `--debug` - postaví aplikaci v debug módu. Implicitně používáno,
- `--release` - postaví aplikaci v release módu,
- `--nobuild` - pouze nainstaluje aplikaci a stavění aplikace úplně přeskočí.

Ukázky příkazu:

```
phonegap run android
phonegap run android ios
phonegap run android --emulator
```

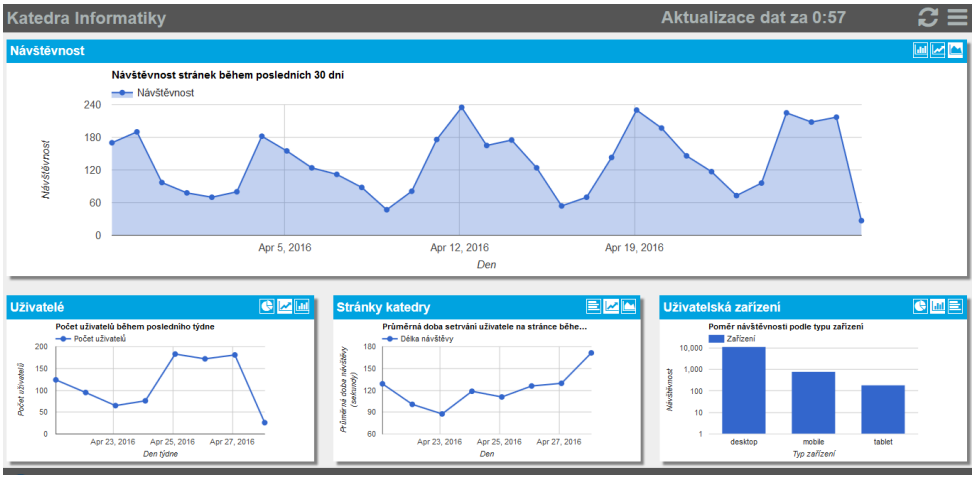
Přestože je PhoneGap aplikace implementována jako webové stránky, nemůže být spuštěna ve standardním webovém prohlížeči, pokud aplikace využívá API zařízení. K otestování, jak aplikace pracuje s jednotlivými API, je potřeba nainstalovat ji na mobilní zařízení nebo ji spustit v emulátoru na počítači.

6.1 Pokročilá správa projektu

PhoneGap Build slouží nejenom jako uložiště pro projekty a jejich automatickou kompilaci v hybridní aplikace. Build představuje vhodné prostředí pro týmy vývojářů, protože umožňuje vlastníkovi projektu pozvat další uživatele ke spolupráci, digitálně podepisovat aplikace a také efektivnější testování. Další informace o rozhraní PhoneGap Build se nachází v příloze.

7 Zhodnocení

Jednoduchými kroky byla vytvořena plně fungující dashboardová aplikace. Zvolením cílového operačního systému je možné vytvořit hybridní aplikaci ve funkční podobě hned pro několik platforem, bez potřeby znalostí nativního jazyka, ve kterém je aplikace zabalena. Obrázek 9 ukazuje, jak vypadá výsledná aplikace ve webovém prohlížeči. Jak je znázorněno v obrázku 9,



Obrázek 9: Výsledná aplikace zobrazená ve webovém prohlížeči

aplikace používá velmi jednoduché prvky, jejichž funkci dokáže uživatel intuitivně pochopit.

Z pohledu funkčnosti se jedná o velmi jednoduchou aplikaci, která pouze zobrazuje data. To nevadí, protože cílem nebylo vytvořit komplikovanou aplikaci, ale popsat proces vývoje aplikace ve PhoneGap. Předností PhoneGapu je, že i když by se vytvářela složitá aplikace, kroky, které je potřeba vykonat ke zprovoznění aplikace na mobilním zařízení, jsou pořád stejné. Z toho vyplývá, že PhoneGap má nízkou křivku zaučení. Vývojář tedy potřebuje pochopit jen několik nastavení ve PhoneGap a zbytek času může věnovat webové části aplikace a jejímu testování.

Problém nastává, pokud se chce vývojář zaměřit i na nepodporované platformy jako je například BlackBerry. PhoneGap dříve podporoval mnoho platforem, ke dnešnímu dni však podporuje pouze iOS, Android a Windows Phone. Pro ostatní platformy je potřeba využívat příkazů cordova a aplikace vytvářet lokálně.

Na druhou stranu, podle statistik na stránkách IDC bylo v roce 2015 na celém světe až 82,8% uživatelů používajících operační systém Android, 13,9% používalo iOS a 2,6% Windows Phone. Ostatní platformy nedosahovaly ani půl procenta. Pokud tedy vývojář nemá značný důvod pro podporu jiných platforem, vystačí pouze s těmi, které PhoneGap podporuje.

8 Závěr

V první části práce byla probrána problematika vývoje mobilních aplikací, jaké problémy jejich vývoj obnáší, jaké typy aplikací se dnes používají a populární nástroje pro vývoj. Dále se tato část zabývala vývojem aplikace v prostředí PhoneGap, přesněji strukturou projektu a možnostmi nastavení projektu. Výsledkem byl projekt, který byl přichystaný pro testování na mobilních zařízeních.

Druhá část práce se věnovala architektuře samotné aplikace. Rozebírala architekturu z několika úhlů. Nejdříve se řešil výběr architektury klienta, následně se řešila architektura uživatelského rozhraní.

Třetí část byla zaměřena na zpracování dat ze serveru. Popisovala, jak se aplikace dotazuje serveru pomocí metody v API od Google Analytics a také jako RESTful dotazem. Poté se zabývala formátem odpovědi serveru a jejího nastavení.

Ve čtvrté části bylo rozebráno testování aplikace. Bylo probráno vše, co je potřeba nastavit, pro spuštění projektu a odzkoušení aplikace v mobilní aplikaci PhoneGapu. Bylo popsáno jakým způsobem testování probíhá a jak se testování spouští. V této části se nacházela také sekce o debugování. PhoneGap debuguje aplikace pomocí rozhraní weinre. Bylo vysvětleno, jak weinre nainstalovat a spustit.

V páté části se řešila kompilace projektu ve výslednou hybridní aplikaci. Probíraly se zde různé možnosti pro vytváření hybridních aplikací, jejich výhody a nevýhody a potřebná nastavení.

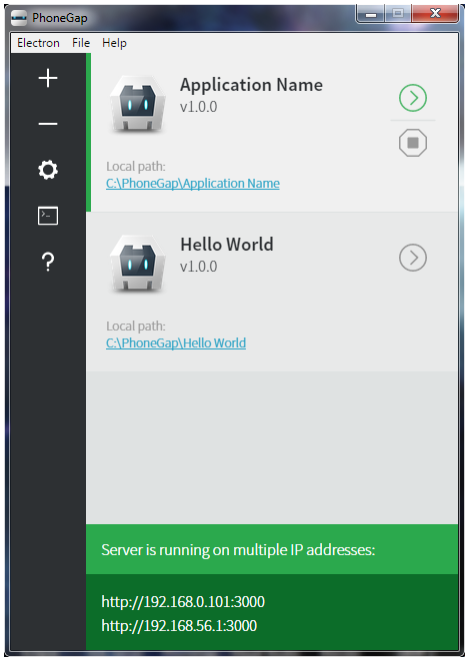
V poslední šesté části se hodnotila výsledná aplikace a možnost pokrytí jednotlivých platforem.

A PhoneGap Desktop Application

Desktopová aplikace je jednodušší na pochopení a pracování než rozhraní příkazové řádky, protože se zde vše řeší graficky přes tlačítka a pomocí rozhraní „drag-and-drop.“ Vše ostatní je pro vývojáře skryté. Oproti CLI ale neumožňuje vytvořit hotovou aplikaci z projektu.

Po instalaci a spuštění aplikace PhoneGap Vás uvítá velmi jednoduché grafické rozhraní. Desktopová aplikace nenabízí nic jiného než je vytvoření, přidání a ubrání PhoneGap projektu a jeho aktivace a deaktivace. Dalšími funkcemi aplikace je pouze nastavení portu pro server a zobrazení server logu.

Při vytváření nového projektu bude PhoneGap vyžadovat `path` (cestu), kde se má nový projekt vytvořit, `name` (název) projektu a `ID`, které představuje unikátní identifikátor. Jako identifikátor je doporučeno používat opačně formátovaný jmenný prostor domény, např. „cz.vsb.dashboard.“ Parametry `path` a `name` jsou povinné.



Obrázek 10: Desktopová aplikace

Přidání projektu je možné přetáhnutím libovolného souboru do okna aplikace, nebo kliknutím na tlačítko „+“ v horní levé části aplikace. Následně se otevře menu pro vytvoření nebo otevření projektu. Při výběru možnosti `Open existing PhoneGap project...` („Otevřít projekt“) je potřeba vybrat umístění projektu. Je zapotřebí vybrat složku, v níž se nachází soubor config.xml. Tento soubor obsahuje název a ID projektu a není možné bez něj projekt do aplikace přidat.

Projekt se aktivuje jeho vybráním a stisknutím tlačítka Play. Pokud je projekt již aktivován, bude levý okraj projektu v aplikaci zvýrazněn zelenou barvou a pod tlačítkem Play bude tlačítko Stop. Je možné mít spuštěn pouze jeden projekt. Při aktivaci projektu se spustí lokální web server se specifickou adresou zobrazenou v dolní části aplikace. V obrázku 10 je první projekt spuštěn a v dolní části obrázku je zobrazena adresa web serveru.

Možnost aktualizace desktopové aplikace se zobrazí automaticky při jejím spuštění. Po kliknutí na tlačítko „Update“ (či „Update now“)se aktualizace stáhne. Po stažení by se mělo kliknout na tlačítko Restart pro načtení aktualizace. Zjištění aktuální verze programu se provádí stisknutím tlačítka „?“ v menu a vybráním možnosti About.

B PhoneGap Build

PhoneGap Build je cloudové uložiště pro kompilaci PhoneGap aplikací a představuje další z možností, jak spravovat projekt a výsledné aplikace. Vytváření aplikace lokálně pomocí mobilní aplikace a CLI je rychlé a snadné. Ovšem je zde i několik problémů. Například mobilní aplikace představuje pouze médium pro spuštění PhoneGap projektu, nikoliv jeho zabalení v hybridní aplikaci. Dalším problémem je, že při lokálním vytváření hybridní aplikaci je potřeba mít k dispozici SDK pro jednotlivé platformy.

PhoneGap Build umožňuje vývojáři svůj projekt nahrát na cloudové uložiště a sám ji zkompiluje v hybridní aplikace pro všechny podporované platformy. Díky tomu naprosto odstraňuje potřebu jakýchkoliv SDK. Umožňuje také zapnout debug režim, nástroj Hydration a režimy pro spolupráci a sdílení projektu. PhoneGap Build umožňuje také aplikaci nechat elektronicky podepsat.

Build neumožňuje projekt vytvářet, pouze projekty nahrávat z lokálního počítače, nebo nahrát projekt z git repositáře.

PhoneGap Build nevyžaduje z lokálního PhoneGap projektu vše - potřebuje pouze obsah složky www zabalený jako *.zip soubor.
Složka www musí mít následující strukturu:

- Soubor index.html musí být v nejvyšší úrovni projektu. Vše ostatní může být na stejné nebo nižší úrovni.
- Soubor index.html musí mít stále přístup k PhoneGap Api, proto musí v hlavičce obsahovat jeden z příkazů ve výpisu 9.

```
<script src="phonegap.js"></script>
<script src="cordova.js"></script>
```

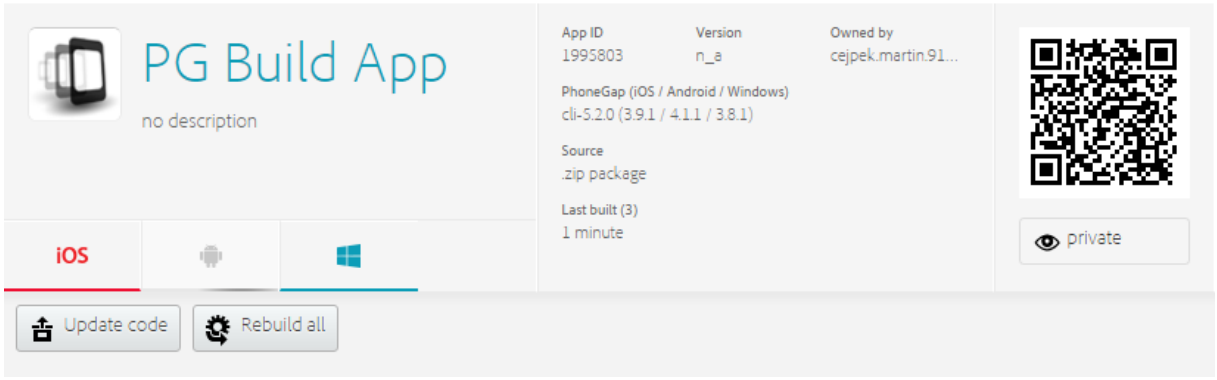
Výpis 9: Navázání na PhoneGap API

- pokud složka www obsahuje phonegap.js nebo cordova.js soubor, je potřeba jej smazat. Build generuje tyto javascriptové knihovny pro každou platformu jinak a pokud by projekt obsahoval nekompatibilní verzi, došlo by k chybě při běhu aplikace.
- Může být přidán i soubor config.xml. Pokud je přidán, musí být pro správné nahrání umístěn na stejné úrovni jako index.html. Pokud připojen nebyl, je možné informace o projektu nastavit ve webovém rozhraní Buildu.
- Je možné přidat soubor `.pgbomit`, který označuje, že daná složka obsahuje soubory, které nemají být přidány do balíčku aplikace.

Po nahrání projektu je možné ještě před jeho postavením zapnout nástroj Hydration, debug režim a v případě, že nebyl přidán/nalezen config.xml, je možné upravit název a popis aplikace.

Po kliknutí na tlačítko „Ready To Build“ se začne projekt předělávat v hybridní aplikaci pro všechny podporované platformy.

Pokud je daná platforma označena červeně, došlo k chybě při vytváření aplikace. Pokud je označená modře, vše proběhlo v pořádku a aplikaci je možné okamžitě stáhnout. V obrázku 11 došlo k chybě při kompilaci aplikace pro iOS. Pro Windows Phone byla postavena úspěšně a pro Android se stále vytváří.



Obrázek 11: Základní rozhraní PhoneGap Build

Po kliknutí na název aplikace se otevře detail projektu. V detailu je možné upravit soubory a nastavení projektu. Jsou zde čtyři záložky:

- Build - zde se spravují aplikace vytvořené z projektu
- Plugins - slouží pro zobrazení informací o pluginech použitých v projektu
- Collaborators - slouží pro správu spolupracovníků
- Settings - zde se nastavuje projekt

B.1 Build

V této záložce se nachází základní informace o projektu a vytváří se zde aplikace pro jednotlivé platformy. Rozhraní zde nabízí pouze možnosti elektronické podepsání aplikace, znovupostavení aplikace pro danou platformu, log a tlačítko pro stažení aplikace. V případě, že při stavění aplikace došlo k chybě, je tlačítko pro stažení nahrazeno tlačítkem pro zobrazení chyby.

Právě elektronický podpis je jednou z nejdůležitějších funkcí této záložky, protože například aplikace pro iOS se bez přidaného elektronického podpisu správně nepostaví. „Elektronický podpis (někdy také digitální podpis) je certifikát, který plně nahrazuje vlastnoruční podpis při elektronické komunikaci.“[7]

Postup vytváření digitálního podpisu je jiný nejen pro každou cílovou platformu, ale i pro každý operační systém ve kterém je vytvářen. Z tohoto důvodu tento proces nebude v této práci dále vysvětlen.

B.2 Plugins

Tato záložka zobrazuje seznam všech použitých pluginů. Název každého pluginu slouží i jako odkaz na stránku s dokumentací k pluginu. Dále se u pluginu zobrazuje zdroj (např. npm), informace o verzi pluginu a na jaké platformy byl použit. Tato záložka je pouze informativní, pro jakoukoliv změnu v pluginech je potřeba upravit samotný projekt.

B.3 Collaborators

Zde se zobrazuje seznam spolupracovníků na projektu.

Přidávání nových uživatelů probíhá formou emailu - pokud uživatel je již registrován pod daným emailem, bude okamžitě přidán do seznamu spolupracovníků. Pokud uživatel registrován není, bude emailem pozván k registraci a v seznamu bude zobrazen jako „nevyřízený.“ Spolupracovníci mohou mít jednu ze dvou rolí:

- Vývojář - může upravovat zdrojový kód projektu a znovu stavět aplikace
- Tester - může shlédnout aplikaci a stáhnout aplikační balíček.

Rozhraní umožňuje kdykoliv uživatele odebrat nebo mu změnit roli, kromě spolupracovníků s označením „nevyřízený.“ Pouze vlastník projektu může spravovat seznam spolupracovníků, smazat projekt a digitálně podepisovat aplikaci.

B.4 Settings

Zde je možné zapnout (případně vypnout) nástroj Hydration, nastavit zdali se jedná o soukromou aplikaci a jestli se má povolit veřejné sdílení. V této záložce je možné nahrát i nové zdrojové kódy a také smazat celý projekt z Buildu.

Pokud aplikace není nastavená jako soukromá, bude považována za veřejnou aplikaci. Veřejné aplikace mají zdrojový kód veřejně přístupný v repositáři GitHub. Veřejné sdílení umožňuje přístup neautentizovaných uživatelů k aplikaci. V nastavení se musí sdílení povolit (tlačítko „Allow public sharing“) a po uložení změn se pod QR kódem zpřístupní tlačítko pro sdílení aplikace.

Nástroj Hydration slouží především pro propagaci aktualizací přímo do aplikací na zařízeních koncových uživatelů. Toho se docílí tím, že zkompiluje nativní binární soubor, kterým „obalí“ mobilní aplikaci. Pokud je nástroj zapnutý a vývojář postaví novou verzi aplikace, bude koncový uživatel (například tester) upozorněn na novou verzi při restartování aplikace.

B.5 Testování aplikace

Aplikace se může zkoušet naskenováním QR kódu v pravé části rozhraní Buildu (viz obrázek 11). Po naskenování bude aplikace projektu stažena a spuštěna v mobilním zařízení. Pod QR kódem je i tlačítko „Install“, které slouží pro stažení a spouštění aplikace v emulátoru na lokální stanici.

C Příloha na CD/DVD

Součástí práce je CD/DVD.

Literatura

[1] LASSOFF Mark. *Mobile App Development with HTML5, LearnToProgram*, 2015, ISBN: 978-0692405055

[2] FERNANDEZ Wilkins. *Beginning App Development with Parse and PhoneGap*, Apress, 2015, ISBN: 978-1484202364

[3] BANGA Cameron. *Essential Mobile Interaction Design: Perfecting Interface Design in Mobile Apps (Usability), Addison-Weslley Professional*, 2014, ISBN: 978-0321961570

[4] BASS L.; CLEMENTS P.; KAZMAN R. *Software Architecture in Practice 2nd Edition Reading*, MA: Addison-Wesley, 2003, ISBN-10: 8177589962

[5] [online] *What is plugin?* URL: $<http://www.computerhope.com/jargon/p/plugin.htm>$ [cit. 2016-03-25]

[6] [online] *What is debugging? - Definition from WhatIs.com* URL: $<http://searchsoftwarequality.techtarget.com/definition/debugging>$ [cit. 2016-03-25]

[7] [online] *Co je Elektronický podpis? | Adaptic* URL: $<http://www.adaptic.cz/znalosti/slovnicek/elektronicky-podpis/>$ [cit. 2016-03-25]

[8] [online] *What is graphical user interface (GUI)? definition and meaning* URL: $<http://www.businessdictionary.com/definition/graphical-user-interface-GUI.html>$ [cit. 2016-03-25]

[9] [online] *Smartphone Average Selling Price (ASP) by OS, USD($), 2012 - 2018* URL: $<http://chartchannel.icharts.net/chartchannel/smartphone-average-selling-price-asp-osusd-2012-2018_m3puyi9fc>$ [cit. 2016-03-25]

[10] [online] *Reasons to changeover to Cross Platform Mobile App Development* URL: $<http://www.oodlestechnologies.com/blogs/Reasons-to-changeover-to-Cross-Platform-Mobile-App-Development>$ [cit. 2016-03-25]

[11] [online] *The New Age of Cross-Platform Mobile App Development* URL: $<http://tech.co/cross-platform-mobile-app-development-2-2015-07>$ [cit. 2016-03-26]

[12] [online] *The Boom in Mobile Apps Continues to Grow - Eazi-Apps Bussiness Opportunity* URL: $<https://eazi-apps-business.co.uk/The-Boom-in-Mobile-Apps-Continues-to-Grow>$ [cit. 2016-03-26]

[13] [online] *What is a Native Mobile App? - Definition from Techopedia* URL: $<https://www.techopedia.com/definition/27568/native-mobile-app>$ [cit. 2016-03-26]

[14] [online] *Ten of the Best Cross-Platform Mobile Development Tools for Enterprises - App Development Marketplace* URL: <*http* : *//appindex.com/blog/ten − best − cross − platform − development − mobile − enterprises/*> [cit. 2016-03-26]

[15] [online] *Native, Web or Hybrid Apps? What's The Difference?* URL: <*http* : *//www.mobiloud.com/blog/*2012/06/*native − web − or − hybrid − apps/*> [cit. 2016-03-26]

[16] [online] *What is native app? - Definition from WhatIs.com* URL: <*http* : *//searchsoftwarequality.techtarget.com/definition/native − application − native − app*> [cit. 2016-04-05]

[17] [online] *HTML5 vs Native Android App | Android Authority* URL: <*http* : *//www.androidauthority.com/html − 5 − vs − native − android − app −* 607214/> [cit. 2016-04-05]

[18] [online] *Framework - Wikipedie* URL: <*https* : *//cs.wikipedia.org/wiki/Framework*> [cit. 2016-04-07]

[19] [online] *A brief history of the mobile app* URL: <*https* : *//www.ding.com/community/history − of − mobile − apps*> [cit. 2016-04-02]

[20] [online] *Four Ways To Build A Mobile Application, Part 3: PhoneGap - Smashing Magazine* URL: <*https* : *//www.smashingmagazine.com/*2014/02/*four − ways − to − build − a − mobile − app − part*3 − *phonegap/*> [cit. 2016-03-26]

[21] [online] *Pros and Cons of Building a Cross-Platform Mobile App* URL: <*http* : *//worryfreelabs.com/pros − cons − cross − platform − app/*> [cit. 2016-03-26]

[22] [online] *PhoneGap Native Plugins | ANDREW TRICE* URL: <*http* : *//www.tricedesigns.com/*2012/03/01/*phonegap − native − plugins/*> [cit. 2016-03-26]

[23] [online] *Hybrid Applications And Android Native Browser | globetrotter* URL: <*https* : *//myshadesofgray.wordpress.com/*2014/04/15/*hybrid − applications − and − android − native − browser/*> [cit. 2016-03-26]

[24] [online] *Hello World Explained | PhoneGap Docs* URL: <*http* : *//docs.phonegap.com/develop/hello − world − explained/*> [cit. 2016-03-26]

[25] [online] *Home · phonegap/phonegap Wiki · GitHub* URL: <*https* : *//github.com/phonegap/phonegap/wiki*> [cit. 2016-03-26]

[26] [online] *The Differences Between Thick & Thin Client Hardware - Webopedia* URL: <*http* : *//www.webopedia.com/DidYouKnow/Hardware*_*Software/thin*_*client.asp*> [cit. 2016-03-26]

[27] [online] *Smart Client Architecture : ASP Alliance* URL: <*http* : *//aspalliance.com/*1169*_Smart_Client_Architecture.*2> [cit. 2016-03-26]

[28] [online] *GUI Architectures* URL: <*http* : *//martinfowler.com/eaaDev/uiArchs.html*> [cit. 2016-03-26]

[29] [online] *Flat Design: An In-Depth Look* URL: <*http* : *//www.awwwards.com/flat − design − an − in − depth − look.html*> [cit. 2016-03-26]

[30] [online] *What Is Flat Design?* URL: <*http* : *//gizmodo.com/what − is − flat − design −* 508963228> [cit. 2016-03-26]

[31] [online] *The beginner's guide to flat design | Creative Bloq* URL: <*http* : *//www.creativebloq.com/graphic − design/what − flat − design −* 3132112> [cit. 2016-03-26]

[32] [online] *20 Beautiful Mobile User Interface For Your Inspiration - Hongkiat* URL: <*http* : *//www.hongkiat.com/blog/mobile − app − ui/*> [cit. 2016-03-26]

[33] [online] *Flat design - Wikipedia, the free encyklopedia* URL: <*https* : *//en.wikipedia.org/wiki/Flat_design*> [cit. 2016-03-26]

[34] [online] *What is a Smartphone? - Definition from Techopedia* URL: <*https* : *//www.techopedia.com/definition/*2977*/smartphone*> [cit. 2016-03-26]

[35] [online] *Smartphone - Wikipedie* URL: <*https* : *//cs.wikipedia.org/wiki/Smartphone*> [cit. 2016-03-26]

[36] [online] *Mobile Operating Systems (Mobile OS) Explained by Webopedia.com* URL: <*http* : *//www.webopedia.com/DidYouKnow/Hardware_Software/mobile − operating − systems − mobile − os − explained.html*> [cit. 2016-03-26]

[37] [online] *Mobile platform Definition from PC Magazine Encyklopedia* URL: <*http* : *//www.pcmag.com/encyclopedia/term/*47144*/mobile − platform*> [cit. 2016-03-26]

[38] [online] *Smartphone sale worldwide 2007 to 2015 | Statistic* URL: <*http* : *//www.statista.com/statistics/*263437*/global − smartphone − sales − to − end − users − since −* 2007/> [cit. 2016-03-26]

[39] [online] *Top Ten Reasons Why People Buy Smartphones* URL: <*http* : *//www.ibtimes.co.uk/smartphone − features − camera − browsing − gaming − apps − gps −* 348123> [cit. 2016-03-26]

[40] [online] *Native, Web or Hybrid Apps? What's The Difference?* URL: <*http* : *//www.mobiloud.com/blog/*2012*/*06*/native − web − or − hybrid − apps/*> [cit. 2016-03-26]